FastMap: Um algoritmo de Indexação, Visualização e Data Mining baseado em Técnicas de Geometria Computacional

Gustavo Tadao Okida e Márcio Katsumi Oikawa
Instituto de Matemática e Estatística
Departamento de Ciência da Computação
Universidade de São Paulo
{gtadao, koikawa}@ime.usp.br

1 Resumo

Uma idéia bem interessante para realizar busca rápida em banco de dados tradicionais e multimídia é mapear objetos em pontos de um espaço k-dimensional fazendo uso de funções de extração de ordem k, fornecidas por um "domain expert" [1]. Podemos, então, utilizar métodos de acesso por região altamente especializados (Spatial Access Method - SAM) para responder a vários tipos de consultas, tais como:

- Busca por exemplo (Query by Example): busca delimitada;
- Todos os pares (all-pairs): proximidade por região;
- Vizinho mais próximo;
- Busca de padrões;
- Busca de pares próximos.

No entando, criar funções de extração não é uma tarefa trivial. É relativamente simples para o "domain expert"estimar a similaridade/distância entre dois objetos. Fornecendo apenas a distância, não é obvio mapear objetos em pontos.

Neste artigo, trataremos com mais detalhes este problema. Vamos descrever um algoritmo rápido para mapear objetos em pontos no espaço k-dimensional, onde k é definido pelo usuário, mantendo a (dis)similaridade. Existem dois benefícios neste mapeamento:

(a) obtenção eficiente dos dados, conjuntamente com o algoritmo SAM;

(b) visualização e data-mining: os objetos podem ser desenhados em 2-d ou 3-d, revelando potenciais "clusters", correlações entre atributos e outras regularidades que o data-mining está procurando.

Introduziremos um método de reconhecimento de padrão chamado Escalonamento multi-dimensional (Multi-dimensional Scaling - MDS). A partir deste algoritmo, proporemos outro mais rápido para resolver este problema, aceitando indexação. Em experimentos reais ou dados simulados, este algoritmo se mostrou mais rápido que o MDS (começa linear, ao invés de quadrático, num banco de tamanho N), enquanto preserva distâncias e toda a estrutura de dados.

2 Introdução

O objetivo deste trabalho é prover uma ferramenta de recuperação e visualização de uma coleção muito grande de dados tradicionais e de multimídia. Uma excelente idéia dada por Jagadish [1] foi ficar dependente de "domain experts"para derivar funções de extração k-dimensaional, como mapear cada objeto em um ponto num espaço k-dimensional. Então o problema ficou reduzido a armazenamento, recuperação e visualisação de pontos, para os quais existe uma grande quantidade de algotimos disponíveis.

No entanto, nem sempre é fácil derivar funções de extração. Considere o caso, por exemplo, de tipo de palavras em inglês, onde a função de distância é a distância de edição (menor número de inserções, remoções e substituições para transformar uma palavra em outra). Não está claro quais mecanismos, ou estratégias, deverão ser utilizados nestes casos. Similarmente, no reconhecimento de padrão de voz, temos tipicamente que fazer algo do tipo *time-warping*, que torna difícil desenhar funções de extração.

Sobrepor estas dificuldades representa um grande desafio. Generalizando a ênfase dada por Jagadish, tentaremos mapear objetos em pontos k-dimensionais, assumindo que o domain expert deve apenas nos fornecer a função de distância/similaridade $D(*,*)$. Note que deve-se ter implementação fácil, por exemplo, o uso da distância Euclidiana entre dois vetores como a função de distância entre os objetos.

Dado um conjunto de objetos e uma função de distância $D()$, são funções desejáveis:

(a) achar objetos similares a um objeto de consulta;

(b) achar pares de objetos que sejam parecidos entre si;

(c) visualizar a distribuição dos objetos em um espaço apropriado, ao invés de verificar "clusters"e outras regras.

A seguir, introduziremos algumas terminologias:

- **Definição 1**: O ponto P_i num espaço k-dimensional, que corresponde ao objeto O_i, será chamando de imagem deste objeto. Ou seja, $P_i = (x_{i,1}, x_{i,2}, x_{i,3}, \ldots, x_{i,k})$.
- **Definição 2**: O espaço k-dimensional que contem as imagens será chamado de espaço alvo ("target object").

A seguir, apresentaremos algumas aplicações que podem utilizar o algoritmo descrito neste trabalho, bem como algumas funções de distância:

- Banco de Dados Multimídia: dada uma coleção de figuras, gostaríamos de achar, dentro de um conjunto, outras similares. Dada uma coleção de imagens, gostaríamos de selecionar outras com mesmo padrão de cores, textura, etc. Aqui utilizamos a distância Euclidiana entre vetores de caracterísicas apropriadas (cor, momentos de inercia por figura, etc). Busca por conteúdo é altamente desejável em banco de dados multimídia, com áudio (voz e música), vídeo, etc. Por exemplo, usuários poderiam querer obter uma lista de músicas ou vídeo clips similares a um padrão escolhido. Uma vez que a função de similaridade foi determinada, o nosso método proposto pode ser imediatamente aplicado;
- Banco de Dados Médicos: onde objetos unidimensionais (eco-cardiogramas), bidimensionais (raio-X) e tridimenionais (tomografia computadorizada) são guardados. Habilidade para recuperar rapidamente históricos com sintomas parecidos pode ser muito importante para um diagnóstigo mais preciso, assim como para o ensino e pesquisa. Note que a função distância é bem complicada, tipicamente requer uma distorção das duas imagens, para certificar que as estruturas (por exemplos os ossos) anatômicas estão apropriadamente alinhadas, antes de consideramos as diferenças. Esta distorção torna difícil a busca por características que descrevem adequadamente cada imagem (e então, mapear isto em pontos num espaço de caracterísitcas);
- Séries temporais, por exemplo: dados financeiros, como valores de ações, número de vendas, etc., ou banco de dados científicos, com

séries temporais de dados sensoriais, de tempo, geológicos, ambientais, astrofísicos, etc. Neste banco de dados, são consultas típicas:

- Ache companhias que possuem valores de ação próximos;
- Ache registro de dias em que o vento magnético solar mostrou padões similares aos de hoje.

O objetivo é ajudar na prevenção, examinando padrões similares que apareceram no passado. Podemos utilizar a distância Euclidiana como função entre duas séries temporais.

- Busca por similaridades em banco de palavras, como no caso de soletrrar, classificar e correção de erros de softwares OCR. Então, dada uma palavra errada, gostaríamos de buscar num dicionário palavras maís próximas desta. Conceitualmente idêntico é o caso de comparação aproximada em banco de dados de DNA, onde existem uma grande coleção dde palavras com quatro letras (ATCG) [2]; a nova palavra tem que ser comparada novamente com as palavras antigas, para achar o melhor candidato. Em todas estas aplicações, a função distância é tipicamente a distância de edição, isto é, o menor número de inserções, remoções e substituições necessários para transformar a primeira palavra na segunda.

- Data Mining e aplicações de visualização. Por exemplo, dado os registros de pacientes (com atributos como gênero, idade, pressão sangüínea, etc.), nós gostaríamos de ajudar os médicos a identificar algum tipo de padrão, ou relação entre os sintomas, dados demográficos e doenças.

Das descrições acima mencionadas, dois tipos de consultas parecem ser bastante desejáveis: "Consulta por exemplo"e "Consulta de todos os pares". Especificamente:

- **Definição 3**: Consulta por exemplo (ou consulta por abrangência) significará consultas do tipo: Dado um objeto desejado (chamado objeto consulta), busque a coleção de objetos mais próximos (dentro de uma determinada distância) do objeto consulta;

- **Definição 4**: Todos os pares (ou busca por região) significará consultas do tipo: Numa coleção de objetos, achar pares de objetos que estão a uma determinada distância entre si.

Todas as aplicações acima mencionadas podem ser vistas como mapeamento dos objetos em algum espaço k-dimensional. O mapemaento trará dois grandes benefícios:

1. Pode acelerar o tempo de busca para as consultas. A razão é que é possível aplicar um refinamento apurado de alto grau chamado *Spatial Access Methods* (SAMs), com árvores R* e ordenação-Z. Estes métodos possuem busca rápida para consultas tanto por abrangência como por regiões.

2. Pode ajudar na visualização, clusterização e mining de dados. Plotando objetos como pontos em 2 ou 3 dimensões podemos revelar muito da estrutura do conjunto de dados, como a existência de clusters maiores, a distribuição geral de figuras (linear x curvilinear x Gaussiana), etc. Estas observações podem ajudar na formulação de hipóteses e descoberta de regras.

Desta forma, o problema genérico é definido abaixo. Podemos referenciá-lo como problema de distância para enfatizar o fato de que a função de distância é conhecida.

- **Problema geral**:

  Dado: N objetos e a informação de distância entre eles (exemplo, uma matriz NxN de distância, ou simplesmente a função D(*,*) entre dois objetos)

  Achar: N pontos num espaço k-dimensional, **tal que** as distâncias são mantidas o mais próximo possível.

Esperamos que a função distância D() seja não-negativa, simétrica e obedeça a inequação de triangularização. No espaço k-d alvo, tipicamente utilizamos a função de cálculo da distância Euclidiana, porque não varia com a rotação das coordenadas.

Um caso especial é quando já conseguimos extrair algumas características, mas ainda estamos interessados em fazer projeções, usualmente porque as características são muitas ("dimesnionality curse"). Nós vamos referenciar isto como caso "Características"

- **Problema especializado:**

  Dado N vetores com n atributos cada;

  Achar N vetores no espaço k-dimensional, **tal que** a distância se mantêm o máximo possível

No problema dado, o mapeamento ideal deveria preencher tais requisitos:

1. Deve ser rápido para calcular: $O(N)$ ou $O(NlogN)$, mas não $O(N^2)$ ou mais alto, porque o custo seria proibitivo para bases grandes;

2. Deve preservar a distância, levando a uma pequena distorção;

3. Deve disponibilizar um algoritmo muito rápido para mapear novos objetos para esta imagem. O algoritmo deve ser $O(1)$ ou $O(NlogN)$. Este requisito é vital para consultas por exemplo.

Na sessão 3 mostraremos uma breve análise do método de Escalonamento Multi-Dimensional (MDS), métodos de redução de dimensão relacionados (K-L, SVD etc.) e apontamentos para literatura de métodos de agrupamento e acesso regional. Na sessão 4, apresentaremos nosso método. Na sessão 5 mostraremos alguns resultados de experimentos em conjutos reais e de testes de dados. Na sessão 6, nossas conclusões.

3 Determinação de Distâncias no k-d Espaço

Apresentaremos, a seguir, formas de tratamento usadas no problema de "distâncias", e os utilizaremos como forma de fundamentar o algoritmo FastMap. Apresentaremos, a seguir a técnica de *Escalonamento Multi-Dimensional*, a transformação *Karhunen-Loève* e a *Decomposição de Valores Singulares*, também usados em problema de redução de dimensionalidade.

3.1 Escalonamento Multi-Dimensional

O algoritmo de Escalonamento Multi-Dimensional (MDS) é utilizado para determinar sub-estruturas, dentro de um espaço dimensional, formadas pelo agrupamento de objetos deste espaço e evidenciem alguma relação entre estes mesmos objetos.

A idéia do algoritmo é minimizar a função *stress*, mapeando cada ponto para um espaço k-dimensional.

$$stress^2 = (\sum_{i,j}(\hat{d}_{i,j} - d_{i,j})^2)/(\sum_{i,j} d_{i,j}^2)$$

onde $d_{i,j}$ é o grau de similaridade entre os objetos O_i e O_j, e $\hat{d}_{i,j}$ é distância euclidiana entre as imagens P_i e P_j. Este algoritmo parte de um valor arbitrário, que é refinado até que se consiga alcançar o menor valor

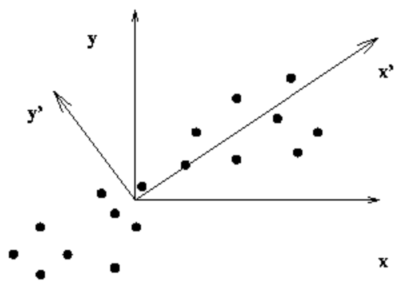

Figura 1: Transformação Karhunen-Loève - o melhor eixo para projetar os pontos é x'

possível. Este valor representa a menor taxa de erro da distância entre objetos nos k-d espaços. Não é objetivo deste trabalho entrar em detalhes deste algoritmo; maiores informações podem ser encontradas em [3].

O algoritmo MDS executa em tempo $O(n^2)$, onde n é o número de objetos do espaço. Apesar polinomial, esta complexidade pode ser considerada alta, uma vez que, na maioria das aplicações de data mining, trabalhamos com um volume considerável de informação. Além disso, devemos considerar que, em nosso caso particular, trabalharemos com consultas dinâmicas, exigindo um tempo adicional para recuperar informação de um novo item de dados do conjunto de dados.

3.2 Técnicas de Redução de Dimensionalidade

No caso 'Características', o problema tem sido amplamente trabalhado, através de técnicas de reconhecimento de padrão estatístico e álgebra matricial. Nosso problema é mapear pontos no espaço n-dimensional para imagens no espaço k-dimensional (k $\leq$ n). O melhor caminho para tratar este problema é através da transformação Karhunen-Loève (KL), que minimiza o erro quadrado significativo, onde erro representa a distância entre o n-d ponto e sua k-d imagem.

A figura 1 ilustra um exemplo de uma transformação KL no plano. Se reduzirmos a dimensão para a reta, a melhor projeção seria obtida na direção x', ou y'.

A transformação KL consiste em um casamento de padrão entre os pontos do n-d espaço. Um algoritmo estatístico é executado e a cada iteração, tentase encontrar um sub-espaço que aproxime a distância entre os pontos. A transformação KL possui, entretanto, algumas restrições:

- Não pode ser aplicado a todos as instâncias deste problema;
- Sempre, no caso 'Características', ele pode ser lento para bancos de dados muito grandes ($N \gg 1$) com muitos atributos ($n \gg 1$).

3.3 Recuperação e Agrupamento

Como mencionado anteriormente, o mecanismo de recuperação será feito pelo método de acesso por região (SAM), capaz de manipular pontos k-dimensionais, retângulos e figuras mais complexas. Existem outras formas de tratar o problema, como árvores-R, árvores-R+, árvores-hB, árvores-R*, algoritmos de ordenação (ordem z) e arquivos *grid*.

Existem também algoritmos de recuperação baseados em desigualdade de triângulos, mas estes não mapeiam objetos em pontos, nem possuem mecanismos de visualização.

Esta abordagem de mapear objetos em k-d pontos pode ser de grande valia para a definição de novos algoritmos de agrupamento (clustering), bem como otimização de algoritmos já existentes. A grande contribuição, em termos de clustering, é sua aplicabilidade, altamente ampla e disponível para vários casos, até mesmo envolvendo um complexo conjunto de dados, tal como GIS (Sistema de Informação Geográfica).

4 Algoritmo FastMap

Para simplificar a notação, adotaremos a seguinte simbologia:
N : Número de objetos no banco de dados
n : Dimensão original do espaço estudado
k : Dimensão do espaço alvo
D(*,*) : Função de distância entre dois objetos
$\|\vec{x}\|_2$: Tamanho do vetor x
(AB) : Tamanho do segment de linha AB

4.1 Algoritmo

O objetivo é resolver o problema para o caso da distância, ou seja, num espaço k-dimensional, achar N pontos onde a distância é dada numa matriz NxN. A idéia é projetar os pontos que estão num espaço n-dimensional num espaço k-dimensional, conhecido. O desafio é calcular as projeções tendo como dado de entrada apenas a matrix de distância NxN.

Deste ponto em diante, um objeto será tratado como um ponto no espaço n-dimensional (n desconhecido).

Dados dois objetos O_a e O_b, escolhidos como pivôs, a idéia central do método é projetar os objetos na linha que passa pelos objetos pivôs no espaço n-dimensional. O algoritmo proposto é chamado escolha_dos_objetos_distantes (), ilustrado mais para frente.

A projeção dos objetos na linha considerada é calculada usando a regra do co-seno (ver figura 2)

Teorema 1 (lei do co-seno): Em qualquer triângulo $O_aO_iO_b$, a regra do co-seno é:

$$d_{b,i}^2 = d_{a,i}^2 + d_{a,b}^2 - 2x_i d_{a,b} \ (2)$$

Prova: Do teorema de Pitágoras aplicado nos triâgulos retângulos O_aEO_i e O_bEO_i, temos que a equação 2 pode ser resolvido para x_i:

$$x_i = (d_{a,i}^2 + d_{a,b}^2 - d_{b,i}^2)/(2d_{a,b}) \ (3)$$

Nas equações, $d_{i,j}$ é a distância D(O_i,O_j) (para $i, j = 1, \ldots, N$). Note que para se calcular x_i é necessário somente a distância entre os objetos, que é fornecida.

Graças a equação 3 é possível mapear objetos em pontos numa linha, preservando algumas das informações de distância. Por exemplo, se O_i é razoavelmente perto do pivô O_a, x_i será pequeno. Com isto, o problema para k=1 foi resolvido.

O problema é verificar se é possível estender este método, pois assim poderíamos mapear os objetos num espaço bi-dimensional e eventualmente num espaço k-dimensional. A resposta é sim, como veremos a seguir. Considere o hiper-plano H, que é perpendicular a linha (O_a,O_b) e projete os pontos do espaço n-dimensional neste hiper-plano. Seja O_i' a projeção de O_i (para $i = 1, \ldots, N$). Agora o problema é o mesmo do problema original, temos que projetar os pontos do espaço (n-1)-dimensional para (k-1)-dimensional.

A única dúvida pode ser como determinar a função distância (D()) entre duas projeções no hiper-plano h, ou seja, O_i' e O_j'. Uma vez feito isto, aplicamos este passo recursivamente.

A figura 3 descreve dois objetos O_i, O_j e suas projeções O_i' e O_j' no hiper-plano H. Uma observação importante é feita no próximo lema:

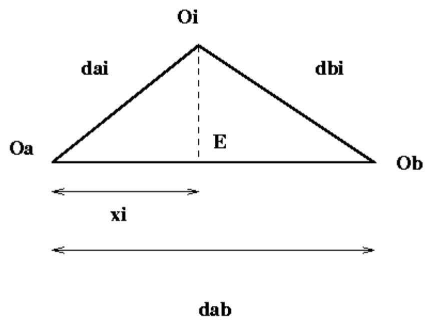

Figura 2: Ilustração da regra do co-seno. projeçao na linha O_aO_b

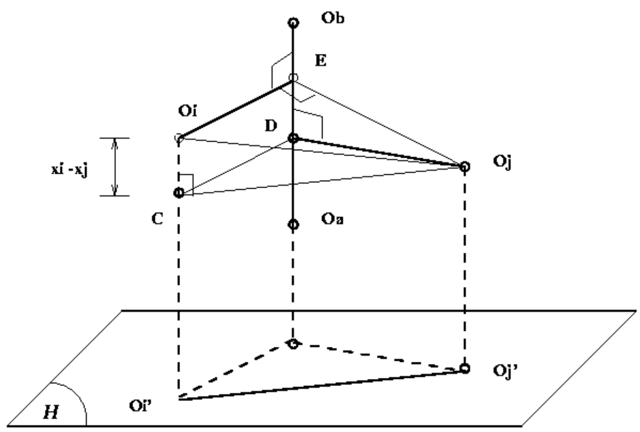

Figura 3: Projeção no hiper-plano H, perpendicular a linha O_aO_b da figura 2

Lema 1: No hiper-plano H, a distância euclidiana D'() entre objetos O_i' e O_j' pode ser calculada através da função d() original da seguinte forma:

$$(\text{D'}(O_i\text{'},O_j\text{'}))^2=(\text{D}(O_i,O_j))^2\text{-}(x_j,x_j)^2 \text{ i,j=1,...,N (4)}$$

Prova: Aplicando o teorema de Pitágoras no triângulo $O_i\text{C}O_j$ temos:

$$(O_i\text{'}O_j\text{'})^2=(\text{C}O_j)^2=(O_iO_j)^2 \text{ - } (O_i\text{C})^2$$

onde (AB) é o tamanho do segmento de linha AB. Desde que $(O_i\text{C}) = (\text{DE}) = \|x_i - x_j\|_2$, a prova está completa.

A possibilidade de calcular a distância D'() nos permite projetar numa segunda linha, sob o hiper-plano H, ortogonal a O_aO_b por construção.

Então conseguimos resolver o problema para um espaço bi-dimensional. O mais importortante é que conseguimos fazer qualquer redução partindo de um espaço k-dimensional.

Um ponto que ainda não foi discutido foi como escolher os pivôs. É importante que esta escolha nos permita ter projeções que sejam o mais distante possível. Para isto, temos que achar objetos O_a e O_b tal que d(O_a,O_b) seja máxima. No entanto, isto custaria O(N^2) cálculos de distância, então foi proposto uma heurística de tempo linear:

Algoritmo escolha_de_objetos_distantes(O, Dist())
Begin
1) Escolha arbitrariamente um objeto e seja ele o segundo obejto pivô O_b
2) Seja O_a = ao objeto mais distante de O_b possível, de acordo com a função Dist()
3) Seja O_b = ao objeto mais distante de O_a possível, de acordo com a função Dist()
4) devolva os objeto O_a e O_b como os objetos escolhidos
End

Todos os passos do algoritmo são lineares. Os passo do meio podem ser repetidos um número constante de vezes, que ainda manterá a linearidade da heurística.

Agora podemos apresentar o algoritmo. De acordo com o problema proposto, o algoritmo recebe:

(a) Um conjunto O de N objetos

(b) A função distância D() que deve obedecer a regra de desigualdade triangular

(c) O número k de dimensões desejados

E devolve:

o mapa dos objetos no espaço k-dimensional, com as distâncias preservadas ao máximo.

O vetor de saída será escrito numa variável global, a matriz $X_{N,k}$. O algoritmo ainda grava quais forma os objetos pivôs para cada chamada recursiva na variável global $PA_{2,k}$.

Algoritmo FastMap

Início

Variáveis globais:

Matriz X[N,k] // A i-ésima linha será a imagem do i-ésimo objeto

Matriz PA[2,k] // Grava os identificadores dos objetos pivôs

Inteiro col# = 0 //Aponta para a coluna de X[] que está sendo atualizada

Algoritmo FaspMap(k, D(), O)

1) Se (K<=0) {retorna}

   senão {col# ++}

2) // Escolha dos objetos pivôs

   seja O_a e O_b o resultado da função escolha_de_objetos_distantes(O, Dist())

3) // Gravação dos objetos pivôs na variável PA[]

   PA[1,col#] = a;PA[2,col#] = b;

4) Se (D(O_a, O_b) = 0)

   faça X[i,col#]=0 para todo i e retorna

5) // Projetar o objetos na linha (O_a,O_b)

   Para cada objeto O_i calcule x_i usando a equação 3 e atualize a matriz global X[i,col#] = x_i

6) // Considere as projeções dos objetos no hiper-plano perpendicular a linha (O_a,O_b); a função distância d() entre dois objetos é dado pela equação 4

   chame FastMap(K-1, D'(), O)

Fim

O Algoritmo calcula as coordenadas de cada objeto na nova dimensão, após cada chamada recursiva do algoritmo. O i-ésimo objeto é mapeado no ponto P_i=(X[i,1], X[i,2], ..., X[i,k] onde X[i,j] é a j-ésima coordenada P_i, a imagem do i-ésimo objeto.

A complexidade do algoritmo é $O(Nk)$ cálculos de distância. Cada chamada recursiva dos passo 2 a 5 leva $O(N)$.

A razão para a qual precisamos gravar os registros dos objetos que foram pivôs em cada chamada recursiva é para facilitar consultas. O algoritmo de busca é o seguinte: quando uma consulta do tipo "busca-por-exemplo"chega, o objeto selecionado O_q está mapeado no ponto k-dimensional do espaço alvo, por projeção dele em linhas apropriadas formada por pivôs específicos, com a função de distância apropriada a cada tempo. Ou seja, repetimos o passo 5 do algoritmo FastMap apenas para consultas.

Note que a complexidade da operação de mapeamento é constante (O(1)) mesmo num bando de dados de tamanho N. Mais detalhadamente, o algoritmo requer θ(k) operações de cálculo de distância, porque precisamos calcular a distância do objeto consultado com cada um dos $2 * k$ objetos pivôs.

5 Discussão e Resultados

Foram feitos dois tipos de comparações:

1. Comparação com uma implementação do algoritmos MDS
2. Visualisação e agrupamento (clusterização) em diversas aplicações

Detalhes dos testes podem ser vistos em [3].

Comparação com uma implementação do algoritmos MDS

Uma implementação do algoritmo FASTMAP foi comparado com outra do MDS e foi melhor já que leva tempo linear contra o tempo quadrático que o MDS leva para variações de N e k. E de acordo com os experimentos feitos, a linearidade do tempo e a boa qualidade dos dados obtidos, o alçgoritmo FASTMAP é uma ótima opção.

Visualisação e agrupamento (clusterização) em diversas aplicações

Foi mostrado também que o algoritmo FASTMAP é uma boa opção para visuaização e agrupamento [3].

6 Conclusão

O algoritmo FASTMAP se mostrou eficiente em mapear objetos em pontos num espaço k-dimensional, preservando ao máximo as características das distâncias entre objetos.

Em outras tentativas de busca por similaridade em bancos de dados não tradicionais ou multimídia, era esperado que uma função de extração fosse definida. Já no FASTMAP o usuário precisa apenas fornecer a função distância D(), a partir da qual o algoritmo irá inferir características apropriadas para cada objeto.

Maper objetos em pontos tem dois objetivos. Primeiro, pode acelerar a busca um vários tipos de consultas (busca por exemplo, todos os pares, vizinho mais próximo, ...) porque vários algoritmos otimizados estão disponíveis (R-arvore, R*-arvore, ...). Segundo, o mapeamento é útil para análise de agrupamento, data mining e visualização de dados multi-dimensionais.

A principal contribuição do artigo [3] é a apresentação do algoritmo FASTMAP, um algoritmo linear que preenche todos o requisitos para o problema de busca de objetos em uma coleção de dados.

1. resolve problemas gerais (do tipo distância) enquanto outros algoritmos como Karhunen-Lòeve (K-L) e a decomposição singular de valores (SVD) pode apenas resolver para casos especiais (do tipo característica);

2. é linear ao tamanho do banco de dados;

3. ao mesmo tempo permite uma indexação muito rápida, sendo possível mapear um objeto novo e arbitrário num ponto k-dimensional em O(k) cálculos de distância, apesar do tamanho N do banco de dados.

O algoritmo utiliza teoremas tradicionais da Geometria (como lei do coseno) e projeta rapidamente cada objeto em uma direção apropriada em cada uma daas k chamadas recursivas. A respeito da qualidade do resultado, os testes comprovam que é muito eficiente.

A segunda contribuição do artigo [FASTMAP] é introduzir ferramentas de reconhecimento de padrões, ciências sociais e algebra de matrizes, e mais espcificamente

Referências

[1] JAGADISH, H. V. A retrieval techniques for similar shapes. *ACM Sigmod Conf.*, p. 208–217, may 1991.

[2] National center for biotechnology information. http://www.ncbi.nlm.nih.org. 1986.

[3] FALOUTSOS, C., LIN, K.-I. D. Fastmap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. *ACM Sigmod Intl. Conf. Management of Data*, p. 163–174, 1995.